ELSEVIER

Computational Geometry 15 (2000) 41–49

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Interactive texture mapping for polygonal models ☆

Hanqiu Sun [a,*], Hujun Bao [b]

[a] *The Chinese University of Hong Kong, Shatin, N.T., Hong Kong*
[b] *State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, P.R. China*

Abstract

Polygonal models have been widely applied in the community of CAD and computer graphics. Since a polygonal surface usually has no intrinsic parameterization, it is very difficult to map textures onto it with low distortion. In this paper, we present an efficient texture mapping algorithm for polygonal models. For each region to be mapped, the algorithm first constructs a B-spline patch with similar shape to surround the model. The mapped region is then projected onto the constructed B-spline patch to achieve a parameterization. By interactively controlling the B-spline patch, the user can conveniently decorate the surface of the model to meet his requirements. Both local and global texture mapping are discussed. The experimental results demonstrate that the algorithm has a great of potential applications in computer animation and virtual reality systems. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Texture mapping; B-spline patch; Polygonal models; Computer graphics

1. Introduction

Due to its simplicity and uniformity, polygonal models have become popular in representing 3D objects in recent years. Since such a model usually consists of a large number of polygons without an intrinsic parameterization, a challenge problem is how to efficiently edit and decorate it in many applications, such as computer animation, virtual reality and visualization. Texture mapping is one of the most successful techniques in computer graphics to decorate the surface of a model, which dramatically enhances the richness of computer synthesized images [9].

Traditionally, a texture map is defined as a map from a 2D image (texture space) to the surface of a 3D model. Each point on the surface is associated with a corresponding pixel in the texture space, and the color of the pixel is used to specify the required rendering attributes. Many texture mapping

☆ This project is partly supported by DG-CUHK/2050197, National Natural Science Foundation of China (Grant No. 69673027), Natural Science Foundation of Zhejiang Province (Grant No. 697011) and Huo Yingdong Young Teacher Foundation.

* Corresponding author.
E-mail address: hanqiu@cse.cuhk.edu.hk (H. Sun).

0925-7721/00/$ – see front matter
PII: S0925-7721(99)00047-4

algorithms have been addressed in two problems. One problem is to construct the texture map with low distortion, and the other is to reduce the texture aliasing. Since a parametric surface is a 2-manifold defined over a 2D parametric space, it becomes trivial if we use the parametric representation of the surface as the map. Early texture mapping algorithms are widely based on this idea [3,5]. Since a polygonal mesh has no such a representation, it is difficult to perform texture mapping by using these algorithms. An efficient texture map is to project the texture onto the surface using an intermediate surface with simple shape that can be efficiently mapped [2]. Due to its independence of the representation of the model, this method has been widely used in computer animation and virtual reality systems. Nevertheless, the method may suffer from high texture distortion. This is because the shape of the intermediate surfaces may be dramatically different with the mapped model, and the map between the model and the intermediate surface cannot correctly reflect its shape. Adhering to this idea, 3D painting method is developed to directly achieve the visual effects of the texture mapping during the interactive phase [8].

Note that with the traditional algorithms, some problems may occur at the junction of two patches [3], and the mapped texture is usually with high distortion due to the non-linearity of the texture map. Many optimization schemes are used to improve the map to get natural mapping [1,7,10–13,15]. The basic idea behind these methods is to minimize the specified distortion energy to relieve the non-linearity of the map. In [1], Bennis et al. presented a flatting algorithm for parametric patches using an incremental relaxation procedure. Maillot et al. described a global texture mapping scheme for polygonal models using energy-minimum technique [13]. Levy et al. proposed an iterative optimization algorithm to assign the texture coordinates to the vertices of the triangulated mesh [10]. Unlike the global optimal methods, this algorithm may efficiently map the texture onto a user-specified region with minimized distortions. Compared with the traditional methods, the optimization schemes can achieve much higher quality of texture mapping with penalty of expensive computation. Once the texture coordinates have been assigned to the vertices, the user is difficult to adjust the mapping results. This disadvantage greatly limits their wide application, because the user need quickly observe the mapped results during interactive design in many applications.

The goal of our work is to achieve an efficient compromise between the traditional methods and the optimal methods. Instead of using simple intermediate surfaces, we use a B-spline patch to roughly fit the surface of the model, the mapped surface is then parameterized by projecting it onto the B-spline patch. Since we can conveniently construct the B-spline representation with lower distortion, the final visual effects are dramatically improved, and the user can control the mapped results by editing the B-spline patch.

This paper is organized as follows. Section 2 outlines the algorithm. The construction of B-spline patch and the parameterization of the model are described in Section 3. Section 4 gives a global mapping method, which can be regarded as an extension of the two-part texture mapping. The seamless mapping strategy is discussed in Section 5. Finally, we conclude our algorithm in Section 6.

2. Overview of the algorithm

As discussed in the previous section, the parameterization is the key problem in texture mapping polygonal surface. From the view of topology, a closed polygonal surface is not homeomorphic to a disk, so it is usually divided into several polygonal meshes with cut boundaries, each of which is mapped

using the constructed parameterization. Of course, in some cases, a closed surface can be regarded as an open surface with common boundaries, and a global parameterization can be conveniently constructed for texture mapping. In the following, we will focus on the 2-manifold polygonal meshes.

Since the connectivity of a polygonal surface may be very complex, it is not a trivial thing to flatten it [6] or to use a parametric patch to interpolate all its vertices [7]. This is the major reason why this kind of algorithms involves expensive computation. As a compromise, our algorithm first constructs a parametric patch roughly following the shape of the polygonal mesh, a map between the polygonal mesh and the constructed parametric patch is then established to achieve the parameterization of the original mesh. Based on this idea, we can outline our interactive texture mapping for polygonal models as follows.

Step 1. Interactively specify a region to be mapped by tagging the vertices.
Step 2. Construct a non-uniform B-spline patch to roughly follow the shape of the polygonal patch in the region.
Step 3. Project the polygonal mesh onto the B-spline patch to evaluate the parameter of each vertex.
Step 4. Establish seamless visual effects between the adjacent polygonal meshes if needed.
Step 5. Interactively adjust the B-spline patch to get the required visual effects.

We will discuss these procedures in details in the following sections.

2.1. Local texture mapping

In this section, without loss of generality, we assume that a quadrangular polygonal mesh is specified to perform texture mapping. For the regions with complex boundaries, some extra vertices are used to construct the polygonal mesh so that the construction of B-spline patch becomes much more efficient and robust. During texture mapping, the extra portion of the B-spline patch is automatically trimmed without any special operation.

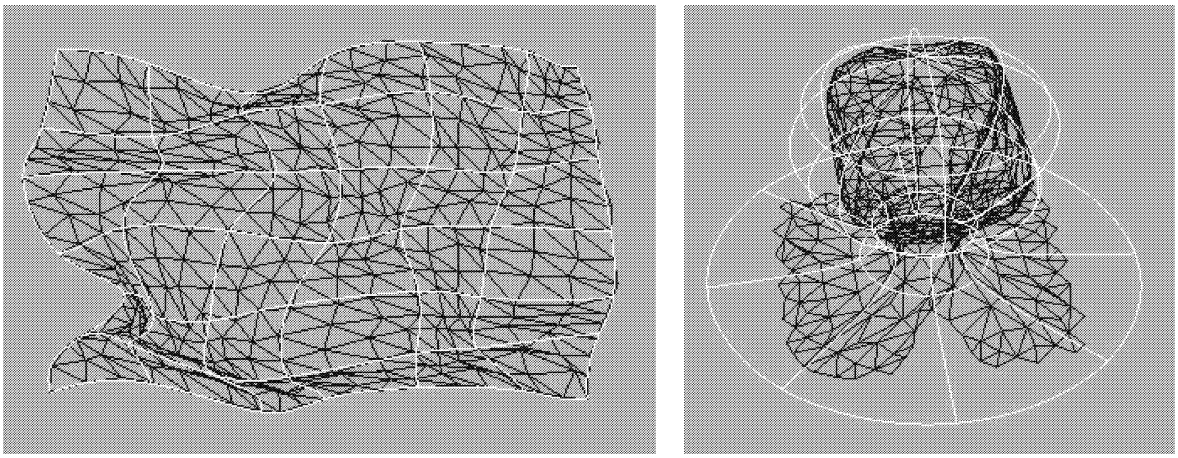
2.1.1. Construction of B-spline patch

Although it is feasible to exactly interpolate the specified quadrangular polygonal mesh is feasible, the complex connectivity of the patch may make the problem difficult. In our application, we just use the B-spline patch to construct and control the parameterization of the polygonal mesh. Therefore, it is enough for us to construct a B-spline patch with a similar shape.

Our algorithm presents two methods to construct the B-spline patch. One is mesh-free method, in which the B-spline patch is constructed using the traditional interactive manner. The user can freely input the control points and the knot vector to achieve the required B-spline patch. The other is mesh-based method, which resamples the polygonal mesh to get a set of rectangular feature points. A B-spline patch is then constructed by interpolating these feature points. Obviously, the second method can fit the shape of the polygonal mesh well with few interactions. In the following, we mainly focus on the second method. We will demonstrate the application of the first method in Section 4.

To simplify the interpolation complexity and reduce the non-linearity of the parametric representation, we just select a small number of vertices as the feature points distributing on the mesh as uniform as possible. The sampling strategy is implemented as follows:

1. Extract the two opposite boundaries of the polygonal patch, and specifying the feature points on the boundaries. For simplicity, the feature points are automatically selected as the vertices distributing on the boundaries with roughly equal distance.



(a) (b)

Fig. 1. The polygonal mesh (in black) and a reconstructed B-spline patch (in white). (a) The mesh-based B-spline patch. (b) The mesh-free patch.

2. Calculate the shortest path on the polygonal mesh between each pair of boundary feature points, and uniformly distribute a user-specified number of feature points on each path.
3. Interpolate the feature points to construct a non-uniform 3×3 B-spline patch.

In the case that the shortest path cannot meet his requirements, the user can interactively edit it. Note that the feature points are not required to strictly locate on the polygonal mesh, they can be freely adjusted after sampling the mesh. Also, the user may modify and extend the B-spline patch in traditional ways to get the required effects. Actually, we just need a B-spline patch with similar shape with the original mesh. Fig. 1 shows a polygonal mesh and a reconstructed B-spline patch. In our current implementation, we use the mesh-based method with small number of feature points to capture the rough shape of the mesh, the mesh-free method is then used to modify the shape to achieve a good match.

2.1.2. Parameterization

The major contribution in our algorithm is to use the B-spline patch to parameterize the polygonal mesh. We use the projection method to achieve the goal. Two projections are provided in the algorithm, one is the parallel projection, and the other is the normal projection.

The parallel projection is to project each vertex of the polygonal mesh onto the B-spline patch along the user-specified direction to achieve its texture coordinates. Let V be a vertex of the mesh, D be the projection direction, a closest intersection point P between ray $r(t) = V + Dt$ and the B-spline patch is first calculated. The parameter (u, v) at point P is then determined in a usual way. Fig. 2(a) illustrates this process. The normal projection is performed in a similar manner. Instead of constant projection direction D, the average normal N of each vertex is used as the project direction to derive the parameterization. Since the normals change over the mesh, the parameterization results of the two methods are different, and the user can select suitable projection mode for different cases. Fig. 2(b) shows the process of the normal projection.

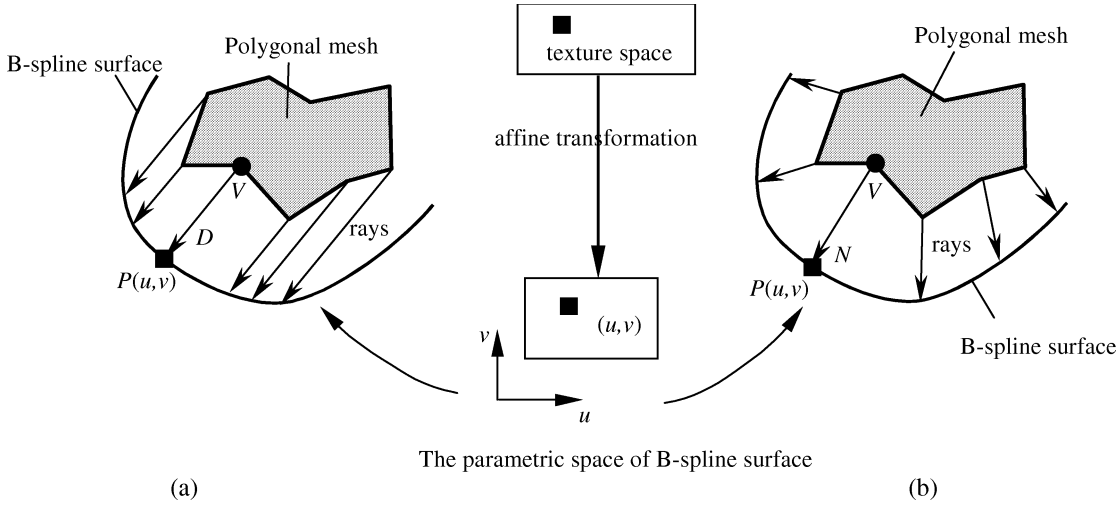


Fig. 2. The two projection modes. (a) The parallel projection. (b) The normal projection.

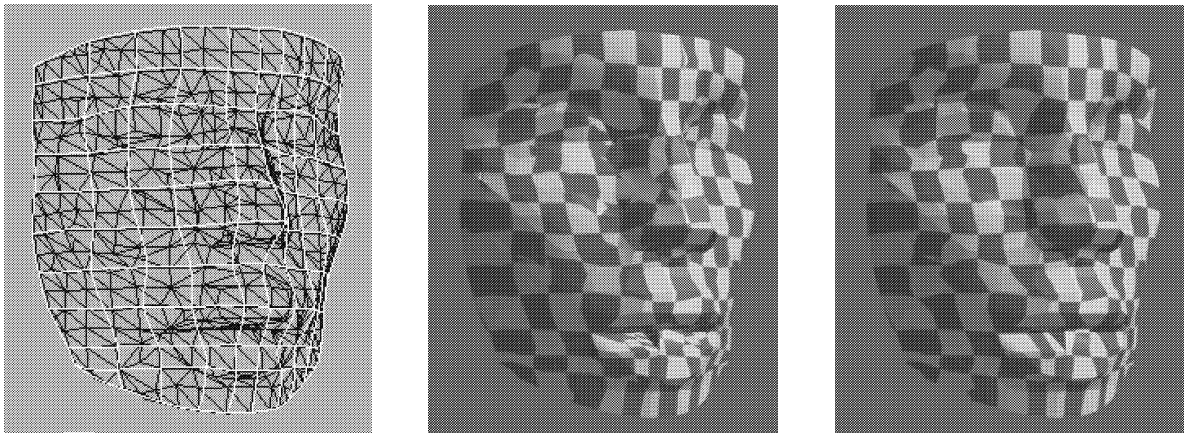
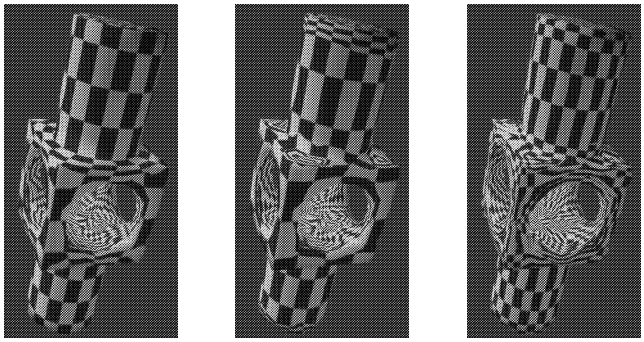


Fig. 3. The mapping results. (a) The polygonal mesh and the reconstructed B-spline patch. (b) The result of the normal projection. (c) The result of the parallel projection.

Note that the parameterization result dramatically depends on the relative position of the polygonal mesh and the B-spline patch. Thus, the user should locate the B-spline patch outside of the mesh and surround the mesh as close as possible. Once the parametric coordinates (u, v) of vertex V are calculated, its texture coordinates can specified by an affine transformation from the parametric space of the B-spline surface to the texture space. As traditional techniques, the affine transformation may be derived using the user inputs, such as rotation angle, translation and tiles, etc. (Fig. 2). Fig. 3 shows the effects with the two projection methods.



(a) (b) (c)

Fig. 4. The comparisons of our method with two-part texture mapping. The normal projection is used. (a) Our method. (b) Two-part texture mapping with cylinder. (c) Two-part texture mapping with sphere.

2.2. Global texture mapping

Obviously, we can also use B-spline patch independent of the polygonal model to perform the texture mapping. Therefore, the algorithm described in the previous section can be regarded as an extension of the traditional two-part texture mapping [2].

2.2.1. Extension of two-part texture mapping

The two-part texture mapping method is a popular method in computer graphics. It uses a simple intermediate surface, such as sphere, cube and cylinder etc., to surround the mapped object. The parameterization is achieved using four kinds of projection modes. Since the shape of the intermediate object may be dramatically different with the mapped object, the texture distortion is usually much high. As a result, the user is difficult to map the texture onto a specified region of the object with low distortion.

Our algorithm can be conveniently extended to deal with two-part like texture mapping. The B-spline patch is used to replace the simple intermediate surfaces. Since the shape of the B-spline patch can be conveniently modified, the texture mapped onto the object is efficiently controlled. Some projection modes used in two-part texture mapping may also be applied in our algorithm. Fig. 4 shows the implementation results with two-part texture mapping and our method. In this example, each vertex normal is calculated by averaging the surrounding triangle normals. Note that at the ends of the mechanical part, the texture distortions are very high because these regions are mapped into the end faces of the cylinder or the pole regions of the sphere. Generally, our method can achieve better texture mapping effects.

2.2.2. Seamless mapping

Note that when the B-spline patch is closed in a direction or adjacent to other surfaces, the texture mapped onto it may have a distinguish seam along the common boundaries. Except for some special cases, the seam seriously affects the visual effects.

Our seamless mapping strategy is to dissolve a small portion of the texture near the common boundary. Without loss of generality, we limit our attention to the closed surface. As shown in Fig. 5(a), the texture

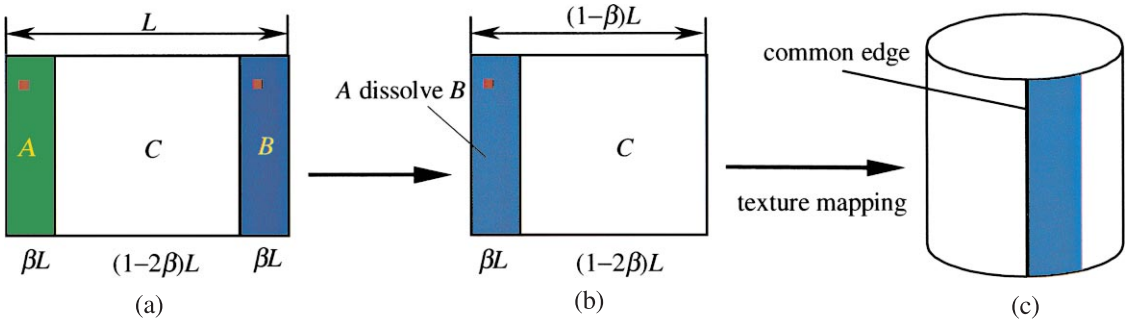


Fig. 5. The seamless mapping. (a) The original texture. (b) The modified texture. (c) The mapping results. The small red squares represent the corresponding pixels.

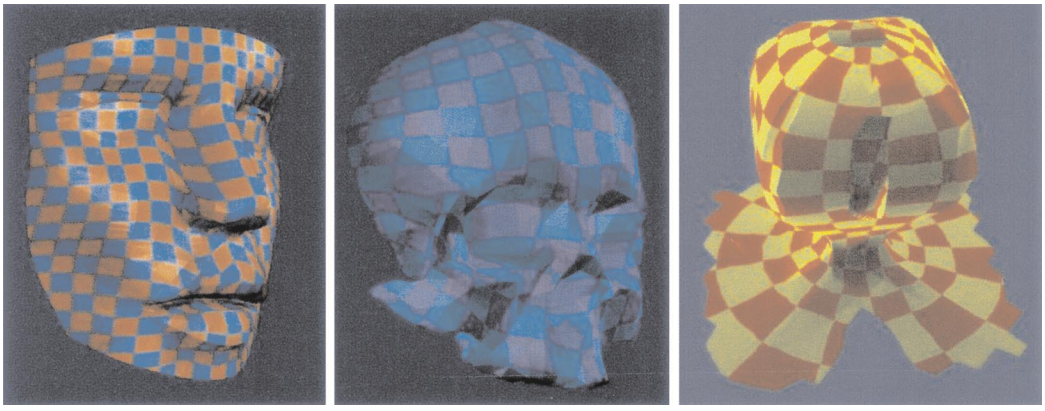


Fig. 6. Texture mapping on three models.

space is divided into three portions A, B and C according to the user-specified percent β. Region C is mapped onto the object as usual, and the regions A and B will be processed before mapped. We use the following equation to blend the texture in regions A and B:

$$I = I_A\big[1 - f(t)\big] + I_B f(t),$$

where I_A, I_B are the colors of the corresponding pixel in regions A and B. I is the color of the pixel after blending.

$$f(t) = \sum_{i=0}^{3} d_i B_i^3(t)$$

is the blending function satisfying $0 = d_3 \leqslant d_1, d_2 \leqslant d_0 = 1$, $B_i^3(t)$ are the Bernstein basis functions. The dissolved texture of A and B are then merged together with region C to form a new texture (Fig. 5(b)). Finally, the new texture is mapped onto the object using the above method. This method can efficiently

relieve the seam visual effects. In practice, β is usually taken as a small fraction, otherwise the texture mapped onto the mesh may have a great change.

3. Implementation and results

We have implemented the algorithm on SGI Octane workstation. Fig. 6 shows the results of applying our method to three models. All the figures are generated using a simple ray casting algorithm. In Fig. 6(a), the model is the same as that in Fig. 3, but with more details. Since the normals on the model change much more smoother than those in Fig. 3, and the constructed B-spline patch fits the original shape more accurately, the texture distortion with the normal projection is greatly relieved. Here, the feature points are automatically determined. Fig. 6(b) is a skull model synthesized using the parallel projection. The B-spline patch is constructed by specifying 36 feature points, and only 16 feature points are located on the model. Fig. 6(c) is a model with two holes. A revolution surface is selected to surrounding the model. The texture is mapped using the normal projection.

Note that we just change the way of specifying the texture coordinates of the vertices, any unaliasing methods, such as mip-map and super-sampling, may easily involve in our scheme. In our current implementation, a jittered super-sampling method is used to generate 9 sample rays in each pixel, and the average intensity is determined for the final display.

The major computation in our algorithm is to calculate the intersection between a ray emitted from a vertex with the B-spline patch. There have many efficient methods to address this problem. In our current implementation, we recursively subdivide the B-spline patch into triangles, and a quadtree with bounding box is used to accelerate the intersection between the ray and triangles. In comparison to the traditional algorithms, our parameterization method is more expensive than the two-part method because of the complex representation of the B-spline patch, but it is much simpler than the optimization methods. Therefore, our algorithm achieves a good compromise between the two kinds of the texture mapping algorithms.

4. Conclusion

We have presented a new texture mapping method for polygonal models. The method is a good compromise between the mapping quality and the mapping control. A B-spline patch is introduced to control the parameterization of the polygonal model. Since the user can interactively modify the shape of the B-spline patch, the texture mapped onto the object is efficiently controlled. The algorithm is also extended to implement two-part like texture mapping and seamless mapping. Compared with the traditional texture mapping methods for polygonal models, our method provides the user more control freedom degree, and the mapping results are usually with low distortion. The experimental results demonstrate that our algorithm has a great of potential applications in computer animation and visual simulation. In our current implementation, when the user modifies the shape of the B-spline patch, the texture coordinates of all vertices of the mesh need to be recalculated. Further work should focus on developing an efficient method to locally update the texture coordinates using hierarchical B-spline editing technique.

References

[1] C. Bennis, J.M. Vezien, G. Iglesias, Piecewise flattening for non-distorted texture mapping, Comput. Graphics 25 (4) (1991) 237–246.
[2] E. Bier, K. Sloan, Two-part texture mapping, IEEE Comput. Graphics Appl. 6 (9) (1986) 40–53.
[3] J.F. Blinn, M.E. Newell, Texture and reflection in computer generated images, Comm. of the ACM 19 (10) (1976) 542–547.
[4] J. Bloomenthal, Modeling the mighty maple, Comput. Graphics 19 (3) (1985) 305–311.
[5] E. Catmull, A subdivision algorithm for computer display of curved surfaces, Ph.D. Thesis, Department of Computer Science of Utah University, December 1974.
[6] M. Eck, T. DeRose, T. Duchamp, H. Hoppes, M. Lounsbery, W. Stuetzle, Multiresolution analysis of arbitrary meshes, in: Proc. of SIGGRAPH'95, 1995, pp. 173–182.
[7] M.S. Floater, Parametrization and smooth interpolation in geometric modeling, ACM Trans. Comput. Graphics 8 (2) (1997) 121–144.
[8] P. Hanrahan, P. Haeberly, Direct WYSIWYG painting and texturing on 3D shapes, Comput. Graphics 24 (4) (1990) 215–223.
[9] P.S. Heckbert, Survey of texture mapping, IEEE Comput. Graphics Appl. 6 (11) (1986) 56–67.
[10] B. Levy, J.L. Mallet, Non-distortion texture mapping for sheared triangulated meshes, in: Proc. of SIGGRAPH'98, 1998, pp. 343–352.
[11] P. Litwinowicz, G. Miller, Efficient techniques for interactive texture placement, in: Proc. of SIGGRAPH'94, 1994, pp. 119–122.
[12] S.D. Ma, H. Lin, Optimal texture mapping, in: EUROGRAPHICS'88, September 1988, pp. 421–428.
[13] J. Maillot, H. Yahia, A. Verroust, Interactive texture mapping, Comput. Graphics 27 (4) 27–34.
[14] S. Marcel, C. Slean, H. Weghorst, Texture mapping and distortions in computer graphics, The Visual Computer 2 (5) (1986) 313–320.
[15] H.K. Pedersen, Decorating implicit surfaces, in: Proc. of SIGGRAPH'95, 1995, pp. 291–300.